Journal of Computer and Knowledge Engineering, Vol. 9, No. 1, 2026. (75-92)

Ferdowsi University of Mashhad

Journal of Computer and Knowledge Engineering

https://cke.um.ac.ir

Information and Communication Technology Association of Iran

Uncertain Virtual Network Embedding Using Neighborhood Information Based Edge Prediction*

Research Article

Arezoo Jahani[1]

10.22067/cke.2025.93455.1159

Abstract *Network virtualization is a key technology for efficient resource sharing in modern data centers, particularly with the advent of paradigms like Software-Defined Networking (SDN) that enable flexible and centralized control. However, virtual network users cannot always express their exact requirements, leading to uncertainties in topology and resource demands. We model such requests as uncertain virtual networks (UVNs). In this paper, we describe such networks as uncertain virtual networks (UVN) and use uncertain graphs to model them. This paper proposes UVNE (uncertain virtual network embedding), which is a three-step algorithm to embed the UVNs. (1) Extracting a certain virtual network from multiple versions of an uncertain virtual network using a proposed edge prediction algorithm based on the SOM (self-organizing map) classifier. (2) Clustering the extracted virtual network with the HCS (highly connected subgraph) clustering algorithm. (3) Embedding the extracted virtual network with a one-step embedding algorithm. The proposed algorithm is compared with edge prediction and virtual network embedding algorithms. The results demonstrate the strength of the edge prediction algorithm, the benefit, and the reduction of the cost of the embedding.*

Key Words Uncertain virtual network, Edge prediction, SOM classifier, Neighborhood information, Graph clustering.

1- Introduction

Modern data center networks (DCNs) face the critical challenge of efficiently managing resources to meet dynamic and diverse application demands. Paradigms such as Software-Defined Networking (SDN) and Network Virtualization (NV) have emerged as foundational technologies to address this challenge. SDN, by decoupling the control plane from the data plane, provides a centralized, programmable, and global view of the network infrastructure. This capability is particularly beneficial for resource allocation tasks like Virtual Network Embedding (VNE), which involves mapping virtual networks (VNs) onto a shared physical substrate. Virtual networks have been created to improve the resource efficiency in the data center networks (DCN). Data center networks consist of a large data center network. Data center networks consist of a large number of servers with communication between them [1]. Virtual networks (VN) are requested as nodes and links with heterogeneous topology [2, 3]. The aim of virtual network embedding (VNE) is to provide the infrastructure with the lowest cost and highest revenue. In the mapping of virtual networks, each virtual node must be mapped onto a physical node with sufficient capacity [4] and each virtual link must be mapped on one or multiple separate parallel paths [5] with sufficient total bandwidth. The responsibility of virtual network embedding is with infrastructure providers (InP). Service providers (SP) are solely responsible for leasing resources from infrastructure providers and providing them to users [6, 7]. Therefore, SPs are connected to users for receiving requests and InP are responsible for providing resources. Virtual network users sometimes have not the ability to express their requirements accurately; (a) the virtual network topology may not be important for them. (b) The nodes' capacity or links' bandwidth may not always be required and their requirements are requested with probabilities on the links. We called such virtual networks as uncertain virtual network (UVN) in this paper. Effectively embedding these UVNs is a complex problem. The centralized control and global visibility offered by an SDN controller make it an ideal platform for implementing advanced UVN embedding (UVNE) strategies, as it can dynamically assess substrate resource availability and make optimal

* Manuscript received May 10, 2025, Revised September 19, 2025, Accepted September 29, 2025.

[1] Associated Professor, Faculty of Electrical and Computer Engineering, Tabriz University of Technology, Tabriz, Iran.

Email: a.jahani@sut.ac.ir

embedding decisions. An example of an uncertain virtual network is shown in Figure 1.

As shown in Figure 1(B), there is an uncertain virtual network with six virtual nodes and seven virtual links. The required capacity of all virtual nodes is shown on nodes and the link bandwidth of all virtual links are shown on links. Also, the uncertain virtual network has probabilities on the links that indicate the amount of need to the link listed in the lifetime of the virtual network. In this paper, we consider the uncertainty on links and do not consider node uncertainty. In fact, we describe the requested virtual networks with uncertain link information as uncertain virtual networks and use uncertain graphs to model them. Since each uncertain virtual network can be converted into several certain virtual networks (CVN), there are two ways to embed uncertain virtual networks in which network topology is not important: (a) Examining all different certain versions of each uncertain virtual network and selecting the best one with the least cost to the user and the highest revenue for service providers. (b) Selecting or extracting a certain virtual network based on neighborhood information or based on the history of virtual networks and then embedding the extracted certain virtual network on the substrate network. The first one creates a complex problem that takes a lot of time to resolve. We use the second method in this article. In this way, at first, we select a certain version of the uncertain virtual network using edge prediction based on neighborhood information and then embed it on a substrate network. The proposed UVNE method has three steps:

- **Step 1.** Edge prediction: this step extracts a certain virtual network (CVN) from multiple certain versions of an uncertain virtual network using a proposed edge prediction algorithm based on the SOM (self-organizing map) classifier. In this step, we need a binary classifier which has been able to train with virtual networks neighborhood information such as; nodes degree and links probability. We proposed SOM-classifier as a binary classifier which is able to organize itself. Also, we train SOM-classifier with neighborhood information. In fact, the training dataset were constructed with (I) running pKwikCluster-EditDistance algorithm on uncertain virtual networks to find the suitable certain version of them. and (II) adding noise to some certain virtual networks.
- **Step 2.** Clustering: this step completes the clustering the extracted certain virtual network with HCS (Highly Connected Subgraphs) clustering algorithm.
- **Step 3.** VNE: this step completes the embedding of the extracted certain virtual network with one-step embedding algorithms.

The proposed UVNE compared with three related uncertain VNE algorithms and the results show that in presence of uncertainty, the cost will be decreased and the revenue will be increased. Because in uncertain virtual networks, we can cluster the network and then embed it on the substrate network. In another test, we examined the strength of the edge prediction algorithm in comparison to other available methods. The results indicate the accuracy of the proposed algorithm.

Our contribution

This paper proposes an uncertain virtual network embedding algorithm which the requested virtual networks are not deterministic and the user's requirement on links, indicate the probability of needing that link over the life of the virtual network. Our contributions are shown in Figure 1. As shown in Figure 1, the requested VN is an uncertain VN and there are multiple certain versions of requested uncertain virtual network. The proposed UVNE, extracts a certain VN (step 1), then clusters the extracted CVN (step 2) and at last, embeds the CVN on a substrate network (step 3).

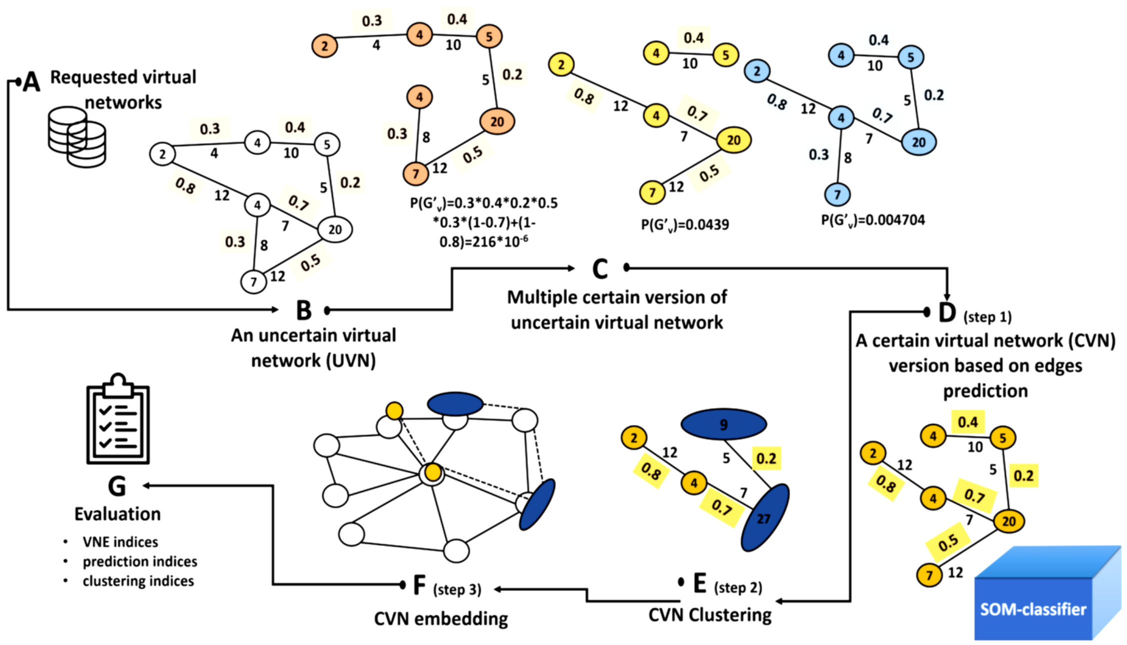


Figure 1. The proposed SOGVNE process

The proposed UVNE framework is designed to leverage the SDN paradigm. The global network view provided by the SDN controller facilitates efficient resource discovery and allocation during the embedding process, particularly in the clustering and VNE steps. Furthermore, once an embedding solution is found, the SDN controller can proactively configure the substrate network (e.g., installing flow rules on OpenFlow-enabled switches) to realize the virtual network, enabling rapid and dynamic provisioning.

The rest of paper is organized as follow. Section 2 expresses the related works. The third section describes the physical and virtual network modelling as well as the concepts of clustering and edge prediction. The fourth section expresses the proposed UVNE algorithm with details. The evaluation steps of the proposed strategy have been discussed in the fifth section and section 6 includes the conclusion and future works.

2- RELATED WORKS

VNE problem is a resource provisioning problem to requested virtual networks. Purpose of most methods is to increase the acceptance ratio. Some VNE methods are resilient in embedding to increase the strength of VNs against failures [8]. Green VNE methods use green energy resources or try to use fewer servers to VNs embedding in each time interval [9], [10], [11]. Topology-aware VNE methods select substrate nodes based on topological information, such as proximity to existing nodes with the high degree or stacking in clusters with identical nodes [12].

In all of the above-mentioned research, VNE has been completed in two phases of node mapping and link mapping [13], [14, 15]. The VNE methods complete these two phases in the uncoordinated or coordinated manner [16]. Coordinated methods are separated into two groups; coordinated two-stage and coordinated one-stage [16]. In the uncoordinated method, two phases are performed sequentially and separately [17]. But in the coordinated method, two phases are performed in synchronization with each other. In the coordinated two-phase, the conditions of the second phase were added to the first phase as constraints, so that the second phase can be verified at this stage to some extent. In the coordinated one-phase, the two phases are completed simultaneously.

A. Uncoordinated methods

In the uncoordinated category, Yu et al. [18] propose a heuristic optimization algorithm for VNE. The main goal in their approach was maximizing the long-term average revenue. They complete node mapping with a greedy algorithm and link mapping with k-shortest path algorithm. Houidi et al. [19] had designed a distributed VNE algorithm responsible for load balancing with a two-stage heuristic algorithm; The first stage is called the selection of hub-and-spoke, that finds the virtual node with the highest capacity as the hub of the cluster and distinguish the neighboring virtual nodes directly connected to the selected hub node to represents the spoke nodes; then removes all hub and spoke nodes and repeats the process to find all hub-and-spokes. The second stage is called mapping of hub-and-spokes, that selects a substrate node with high remained capacity and embeds the first hub on it and the selected node should be mapped all itself spokes on the substrate network and repeats this process for all available hubs. Botero et al. [20] introduce hidden hops constraints. They believe each virtual link which is embedded on a substrate path, needs the CPU demand in the intermediate nodes (hidden hops) of the substrate path and this problem has not been taken into account as an important parameter in related works. To address this problem the authors of [20] add hidden hops constraint to a heuristic algorithm in their proposed method. In Ref. [21] VNE problem was formulated as a mixed integer linear programming problem and was introduced stochastic bandwidth demand to solving it. Most related works complete node mapping phase with node ranking based on their remained capacity. But Ref. [22] uses node connectivity between each pair of nodes which was formulated to measure the resource ranking of the nodes as a new feature to node ranking.

In VNE problem, physical node reusable mapping (PNRM) [23] shows a virtual node should be mapped on a single physical node and all pair virtual nodes of same VN should be mapped on separated substrate nodes. But Ref. [24] to reach high acceptance ratio and high revenue, ignored PNRM and has maps one virtual node in several substrate nodes. Mijumbi et al. [25] use multi-agent for resource management in data centers. All of the nodes and links have an agent which agents can learn environment status and update their policy to do the best action in the future. The authors use q- learning method for agent training. Gong et al. [26] propose a new metric, i.e., global resource capacity (GRC) which computes the embedding potential for each substrate resources. Wang et al. [27] proposed Presto as a VNE method. Presto solves VNE in a preemptive strategy. They convert the substrate network to a tree by Blocking Island (BI) method, which can select suitable substrate nodes and paths to virtual nodes and links in O(1). But recording the extracted tree needs space and the constructed tree should be updated with any changing in the substrate network. Haeri et. al [28], used Markov Decision Process (MDP) framework to formulate node mapping phase and solve it by Monte Carlo tree search algorithm.

Some papers in order to reduce VNE time, complete VNE in a distributed way. Ref. [29] proposed a distributed embedding algorithm with enabling the embedding of multiple VNs in parallel. To reach the specified aim, the authors partition the physical network hierarchically. The first layer includes all physical network and the last one includes some clusters of nodes. VNE starts from the last layer and each VN assigns to a cluster. If the assigned clusters could embed the VNs, repeat this process and take other VNs. But if each cluster could not embed VN, send it to the previous layer which has more resources and this process continue to VN embedded successfully.

Botero et al. [30] proposed an energy efficient VNE, where the objective is to switch off as many substrate nodes and interfaces as possible by allocating the virtual demands to a consolidated subset of active physical networking equipment. Guan et al. [31], proposed an energy efficient VNE by future migration which plans in embedding time. They consider the amount of used energy at day and night and embed a VN at day on suitable substrate resources and plan to embed the VN at night on the other resources to

reduce used energy.
Oliveira et al. [32], proposed an opportunistic resilience embedding (ORE) algo- rithm to embed VNs and protect VNs against substrate network disruptions. Their proposed algorithm has a two-fold: a proactive strategy and a reactive strategy. The proactive strategy smoothes the initial disruption and its impact and the reactive strategy recovers any disruption in the resources.

B. Coordinated two-phase methods

In the coordinated two-phase manner, Chowdhury et al. [33] used integer programming to formulate VNE problem. Then propose two algorithms to relax integer programming constraints: a deterministic and a randomized. In deterministic VNE, they extend the physical network graph by introducing meta nodes for each virtual node and then treat with each virtual link with bandwidth constraints as a commodity consisting of a pair of meta-nodes and solve it with the multi-commodity flow problem. Cheng et al. [34] proposed two algorithms. The prime proposed algorithm uses node ranking concept to map virtual requested nodes on substrate nodes, then embeds the virtual links by the shortest path and multiple commodity flow problems with splittable paths. The second proposed algorithm is a backtracking VNE based on breadth-first search, which embeds both virtual nodes and virtual links in the same stage. Zhang et al. [35] solved VNE by particle swarm optimization (PSO) meta-heuristic and to reduce the time complexity of the link mapping stage. proposed a new node ranking approach to made a balance between used energy and reached revenue.

Hesselbach et al. [36] used the paths algebra-based strategy by coordinating, in a single stage. They map nodes and links with a node ranking method which made of the bi-directional pair of nodes of the physical network and ordered by their available resources. Ref. [37] proposes a heuristic VNE method to minimize the total physical resources required when the virtual network operators request substrate resource shar- ing among multiple priority classes within their virtual networks. Bienkowski et al. [38] used migration when service access position changes. Their main contributions are a randomized and a deterministic online algorithm that achieves a competitive ratio of O(nlogn) in a simplified scenario, where is the size of the physical network. Butt et al. [39] present a mechanism to differentiate among resources based on their importance in the substrate topology and also proposed a set of algorithms for re-optimizing and re-embedding initially-rejected VNs. Soualah et al. [40] used Gomory-Hu tree to solve the VNE which is formulated as an integer linear program.

C. Coordinated one-phase methods

The last category is coordinated one-phase which completes the embedding process in only one phase (node mapping and link mapping complete concurrently) [41]. Ref. [42] was the first one-phase method and it detected subgraph isomorphism with requested virtual network. Houidi et al. [43] proposed a one-stage algorithm with split- ting ability of the virtual network provisioning request across multiple infrastructure providers and solving it using both max-flow min-cut algorithms and linear program- ming techniques. Fajjari et al. [44] proposed a Max-Min ant colony meta-heuristic VNE approach. Botero et al. [45] introduced the energy-aware VNE. Pages et al. [46] introduced the VNE for optical networks in only one phase.

Yu et al. [47] suggested one step VNE that increases coordination. They were selecting some candidate nodes and then select nodes with enough link bandwidth to embed VNs. Ref. [48] accelerated the convergence of PSO VNE meta-heuristic with topology-aware node ranking. Ref. [49] coordinated VNE reducing the number of backtracks by carefully choosing the first virtual node to map. Ref. [50] used integer linear programming problem and multicommodity flow allocation problem respectively for the node and link mapping. Zhu et al. [51] suggested an ant colony based VNE which complete node and link mapping stages concurrently.
Due to recent advances in the field of VNE, deep learning and artificial intelligence-based approaches have attracted much attention. For example, [16] presents an advanced framework for optimal virtual network embedding by combining graph attention network (GAT) and deep reinforcement learning (DRL). Similarly, [25] uses graph convolutional networks (GCN) to assist the embedding process. Also, [12] proposed a multi-criteria fuzzy inference system to solve the virtual network function placement problem (VNFP). Although these methods are powerful, they are mainly focused on deterministic virtual networks. This paper aims to fill this gap by providing a specialized solution for uncertain environments where complete topology information is not available. We will also compare our method (UVNE) with simulations of these new approaches in the experimental section.

TABLE 1
Frequently used notations

Notation	Notation Description
G_s	Substrate network
N_s	Set of substrate nodes
E_s	Set of substrate links
$C_s(n_s)$	Capacity of substrate node $n_s \in N_s$
$B_s(e_s)$	Bandwidth of substrate link $e_s \in E_s$
P_s	Set of all substrate path
$B(p)$	Bandwidth of substrate path $p \in P_s$
G_v	**Virtual network**
N_v	Set of virtual nodes
E_v	Set of virtual links
P_v	Set of existence probability of virtual links
$C_v(n_v)$	Required capacity of virtual node $n_v \in N_v$
$B_v(e_v)$	Required bandwidth of virtual link $e_v \in E_v$
$Y_{n_s}^{n_v}$	Virtual node n_v mapped on substrate node ns (if mapped will be *1* and otherwise will be *0*)
$X_{e_s}^{e_v}$	Virtual link e_v mapped on substrate link es (if mapped will be *1* and otherwise will be *0*)

3- PROBLEM FORMULATION

The proposed UVNE algorithm operates within a broader Software-Defined Networking (SDN) framework, as illustrated in Figure 1. The SDN controller acts as the infrastructure provider (InP), maintaining a global and real-time view of the substrate network resources, including the available CPU capacity of nodes and bandwidth of links. This global view is crucial for the UVNE process. When an uncertain virtual network (UVN) request arrives, the SDN controller invokes the UVNE algorithm. The algorithm's three steps—edge prediction, clustering, and embedding—utilize this centralized knowledge base to make informed decisions. After a successful embedding solution is computed, the SDN controller translates it into specific configuration commands (e.g., installing flow rules via the OpenFlow protocol) to program the underlying physical switches and servers, thereby instantiating the virtual network. This integration enables dynamic, efficient, and automated provisioning of uncertain virtual networks.

This section formulates substrate/infrastructure network, uncertain virtual networks and used graph clustering algorithms.

A. Substrate network

Let $G_s = \{N_s, E_s\}$ be a substrate network, where $N_s$is a set of nodes and E_s is a set of edges. Substrate network has two attributes; where $C_s(n_s)$ is the capacity of substrate node n_s and $B_s(e_s)$ is the link bandwidth of substrate link e_s which is shown in Tabb 1. In substrate network, there is multiple path ($p \in P_s$) and P_s is substrate path set. $B(p)$ is bandwidth of path p which is minimum bandwidth of all links in path p.

B. Uncertain virtual network

Let $G_v = \{N_v, E_v, P_v\}$ be a uncertain virtual network, where N_v is a set of nodes and E_v is a set of edges. P_v is a set of the existence probability of virtual links. So $P_v : E_v \rightarrow (0, 1]$ and assigns a probability between zero and one to all virtual links which shows the existence of that virtual link in network. Uncertain virtual network has two attributes; where $C_v(n_v)$ is the capacity of virtual node nv and $B_v(e_v)$ is link bandwidth virtual link ev which is shown in Table 1.

We formulate uncertain virtual network (UVN) by uncertain graph. Therefore, each UVN has multiple certain version of virtual network. As shown in Figure 1(C), the UVN has multiple certain VN which three of them demonstrated. So, a UVN is also called a probabilistic VN. Suppose a probabilistic/uncertain VN G_v as a universal set, where includes multiple deterministic/certain version of G_v like $G'_v \sqsubseteq G_v$. A deterministic/certain VN G'_v can be considered to be an instance of G_v, according to the probability distribution $P\,(G'_v)$. The probabilities assigned to the VN's links are treated as mutually independent variables. Assuming independence among links, the probability distribution over discrete certain VN is computed as Eq. 1.

$$P\,(G'_v|G_v) = \prod_{e_v \in E_v} P_v(e_v) \prod_{e_v \in E \setminus E_v} 1 - (P_v(e_v)) \quad (1)$$

C. Virtual network embedding

As mentioned before, virtual network embedding process has two phases of node map- ping and link mapping. In node mapping phase, each virtual node should be mapped on a substrate node with adequate capacity. We formulated this concept with a binary variable; $Y_{n_s}^{n_v}$, where shows virtual node n_v is mapped on substrate node n_s or not (If it mapped, then $Y_{n_s}^{n_v} = 1$ and otherwise $Y_{n_s}^{n_v} = 0$). Also a node mapping is possible, when the virtual node's requested capacity is equal or less than the remained substrate node's capacity ($C_v(n_v) \leq C_s(n_s)$).

In link mapping phase, each virtual link should be mapped on a or multiple path with adequate bandwidth as shown in $X_{e_s}^{e_v}$ where demonstrate virtual link e_v is mapped on substrate link e_s or not (If it mapped, then $X_{e_s}^{e_v} = 1$ and otherwise $X_{e_s}^{e_v} = 0$). A link mapping is possible, when the virtual link's required bandwidth is equal or less than the remained bandwidth of substrate path ($B(e_v) \leq B(p)$).

D. Uncertain graph clustering

In this section, we consider the problem of uncertain graph clustering. We formulate both substrate and uncertain virtual networks as graph in this paper and as we know the uncertain graphs also called probabilistic graphs. In this way, at first we consider EditDistance in uncertain graphs and then consider pKwikCluster algorithm to extract all clustered uncertain graph.

Definition 1 (EditDistance). Let G and Q be two deterministic graphs, where $G = (V, E_G)$ and $Q = (V, E_Q)$, we define EditDistance between G and Q as the number of edges that need to be added or deleted from G in order to be transformed into Q which is shown in Eq. 2.

$$D(G, Q) = |E_G \setminus E_Q| + |E_Q \setminus E_G| \quad (2)$$

If G and also Q denoted by adjacency matrix (includes 0 or 1), So Eq. 2 can be written like Eq. 3.

$$D(G, Q) = \frac{1}{2} \sum_{x=1}^{n} \sum_{y=1}^{n} |G(x, y) - Q(x, y)| \quad (3)$$

Kollios et al. [52] extend EditDistance definition between a deterministic and an uncertain/probabilistic graph.

Definition 2. Suppose we have a deterministic graph $Q = (V, E_Q)$ and a uncertain graph $G = (V, E_G, P_G)$. The EditDistance between G and Q is defined as expected EditDistance between all certain version of G like $G' \sqsubseteq G$ and Q that is shown in Eq. 4.

$$D(G, Q) = \sum_{G' \sqsubseteq G} P(G') D(G', Q) \quad (4)$$

In Eq. 4, $P\,(G')$ is probability of deterministic graph G' and is computed by Eq. 1. $D(G', Q)$ is EditDistance

between two deterministic graphs G' and Q that is computed by Eq. 3.

Definition 3 (Cluster graph). A cluster graph $C = (V, E_C)$ is a deterministic graph with the following properties [52]:

1. C defines a partition of the nodes in V into k parts, $V = \{V_1, \dots, V_k\}$ such that $V_i \cap V_j = \emptyset$.
2. For every $i \in \{1, \dots, k\}$ and for every pair of nodes $v \in V_i$ and $v' \in V_i$, we have that $\{v, v'\} \in E_C$.
3. For every $i, j \in \{1, \dots, k\}$ with $i = j$ and every pair of nodes v, v' such that $v \in V_i$ and $v' \in V_j$, $\{v, v'\} \notin E_C$.

Definition 4 (pKwikCluster algorithm). pKwikCluster algorithm [53] is a clustering algorithm for uncertain graphs which solve the problem "Finding a cluster graph $C = (V, E_c)$ for a probabilistic graph $G = (V, P)$ such that $D(G, C)$ is minimized." pKwikCluster algorithm starts with a single node and then find all neighbor nodes of selected node with probability higher than p (a constant variable) and adds all of them to a cluster. If there was not any neighbor with this status, the selected node adds to a single cluster and algorithm continue with other nodes until all nodes consider. Algorithm 1 shows the pKwikCluster algorithm.

Algorithm 1 pKwikCluster Algorithm [52]

repeat
  Choose $u \in V$ randomly;
  $C(u) \leftarrow u$;
  for $v \in V$ such that $p(u, v) > 0.5$ do
    $C(u) \leftarrow C(u) \cup v$;
  end for
  $V \leftarrow V - C(u)$;
until $V = \emptyset$

E. Certain graph clustering

The process of dividing a set of input data into possibly overlapping, subsets, where elements in each subset were considered related by some similarity measure is called clustering. In deterministic graphs, dividing nodes to multiple clusters called graph clustering. Between-graph clustering methods divide a set of graphs into different clusters and within-graph clustering methods divides the nodes of a graph into clusters. In this paper we need standard within-graph clustering methods which are: k-spanning tree, shared nearest neighbor, betweenness centrality based, highly connected components, maximal clique enumeration, and kernel k-means.

The HCS (Highly Connected Subgraphs) clustering algorithm [54] is an algorithm based on graph connectivity for cluster analysis, by first representing the similarity data in a similarity graph, and afterwards finding all the highly connected subgraphs as clusters. This algorithm finds min-cut of mentioned graph and if min-cut of graph was less than |V |/2, then divide graph by min-cut and repeat the process of cutting on two separated subgraphs and otherwise finish the algorithm process and return the extracted subgraphs. HCS graph clustering is shown in Algorithm. 2.

Algorithm 2 HCS clustering Algorithm \\ Highly connected Subgraph

function HCS(G(V,E))
if G is highly connected **then**
  return (G);
else
  (H1,H2,C) ← Min − cut(G);
  HCS(H1);
  HCS(H2);
end if
end function

4- PROPOSED UVNE ALGORITHM

This paper proposes UVNE as a new method for uncertain virtual network embedding. In this paper, we formulated uncertain virtual networks which do not have certain link information by uncertain graph and called these types of virtual networks as uncertain virtual network. The proposed UVNE has three steps; In the first step, the best certain virtual network (CVN) extracted from requested UVN by a proposed SOM-classifier. In the second step, it uses the HCS algorithm to clustering the extracted CVN. Then in the third step, it uses a VNE algorithm to embed CVN on the substrate network. The proposed UVNE algorithm is given in Algorithm 3.

Algorithm 3 UVNE algorithm // **U**ncertain **V**irtual **N**etwork **E**mbedding

Require: *input*: G_s (Substrate Network State from SDN Controller); G_v (Uncertain VN Request);
output: embedding results in X and Y;
G_v=**SOM-classifier**(G_v);
G_v=**HCS**(G_v);
(X, Y) =**VNE**(G_v, G_s);

A. First step: Edge prediction

In the first step, the proposed SOM-classifier applies to extract suitable certain virtual network from multiple certain versions of requested virtual network. In fact, proposed SOM classifier presented to predict edges existence of uncertain virtual network based on neighborhood information. We extract five various graph neighborhood information to train the SOM classifier and use the same classifier to predict the edges of uncertain requested VNs. This step converts an uncertain (probabilistic) virtual net- work to a certain (deterministic) virtual network based on predicting the existence of the link in uncertain VN based on five neighborhood information (see section IV.A.1).

Traditional works in the uncertain graph, applying a threshold and delete all edges lower than a threshold to make a certain graph or compute the probability of all possible certain graphs and select the best certain graph with high probability. However, in uncertain virtual networks, although network topology is not important,

the performance of network and its responsiveness is very important at the times required. So, the threshold-based approaches are not suitable in uncertain virtual networks. In order to address this problem, the current algorithm (SOM-classifier) presents an intelligent approach instead of threshold-based approaches. It uses a binary classifier based on supervised self-organizing map neural network to predict the existence of edges.

At first, the SOM-classifier is trained using certain virtual networks which were collected as training dataset by applying the pKwikCluster algorithm on uncertain VNs and applying noise to various certain VNs. Then, the same classifier is used to predict the existence of edges in UVN. The proposed classifier will be tested before using a natively uncertain virtual network which are generated by GT-ITM tools [55] and the comparison is done by Davies-Bouldin index (DBI), Dunn index (DI), and Silhouette coefficient (SC) metrics. The proposed SOM-classifier is given in Algorithm. 4.

As shown in Algorithm. 4, at first the SOM-classifier trained and tested by the collected dataset (rows 2-3). Then, the extracted features which are extracted from neighborhood information (row 6) (section 4.1.1) are applied to have trained SOM- classifier (row 7) to get a certain virtual network (row 8). The details of the proposed algorithm will be explained in the next subsection.

The edge prediction step, while primarily analyzing the UVN request itself, can be significantly enhanced by the global network visibility provided by the SDN controller. In a practical SDN-based implementation, the controller's centralized monitoring capabilities can collect rich historical data on previously embedded virtual networks. This includes real-world metrics such as actual link utilization, traffic patterns, and the lifespan of virtual links. Such data can be used to refine the training dataset for the SOM-classifier, moving beyond purely topological features toward *performance-aware* link prediction. For instance, links that were frequently utilized or critical to network performance in past embeddings can be weighted more heavily during training. Although the current model relies on features extracted from the UVN topology, the SDN framework paves the way for future extensions where features like link centrality (CEN) could be dynamically computed based on the current substrate network state, leading to even more accurate and context-aware predictions.

I. Feature extraction

In an uncertain virtual network, there are several attributes for an uncertain link, such as the probability of mentioned link and neighbor links, end nodes degree, graph connectivity and the number of the shortest path passing through the mentioned link. Complete knowledge of each uncertain virtual network is crucial for training the SOM- classifier and provides complete information from the uncertain virtual network. We extract five features for every uncertain link and use them to train the SOM-classifier. In fact, the classifier is a binary classifier and should answer the mentioned link will be required in the requested virtual network or not and this answer completely depends on extracted features. So, the SOM-classifier predicts the existence of edges in an uncertain virtual network and extract the suitable certain virtual network.

Algorithm 4 SOM-classifier algorithm

Require: *input*: G_v, TrainingSet, TestingSet, numEpoch (Number of epochs), η (Learning rate);
output: G_v (extracted certain virtual network);

1. make **SOM-classifier**;
2. **Training** (SOM-classifier, TrainingSet, numEpoch, η);
3. **Testing** (SOM-classifier, TestingSet);
4. **Loop**
5.   **if** $E_v \neq \emptyset$ **then**
6.   **for** $e_v \in E_v$ **do**
7.   *FeatureVector*=getFeatureVector(e_v);
8. G_v=SOM-classifier.getOutput(*FeatureVector*);
9.   return G_v;
10.   **end for**
11.   **end if**
12. **end loop**

The extracted five features for each uncertain link (e_v) is listed as below:

1. **Link probability** (P_v): the probability of an uncertain link e_v in an uncertain virtual network G_v is $P_v(e_v)$ and has a large impact in predicting uncertain link hesitance.
2. **Average of neighbor links probability** (AVG_{P_v}): the average of neighbour links probability which is computed by Eq. 5, considers the neighbor links existence probability to decide about the existence of the mentioned link.

$$AVG_{P_v} = \sum_{e_v \in E_v} P_v(e_v)/|E_v| \tag{5}$$

3. **Sum of end nodes degree** (DEG): each uncertain link has two end nodes degree and this feature computes the sum of end nodes degree.
4. **Graph connectivity** (CON): This feature considers the graph's connectivity in the absence of a link. If in the absence of the link, the graph is connected, it receives a zero value. Otherwise, it will receive a value of one.
5. **Link centrality** (CEN): centrality is a concept which is computed how many paths will pass through the mentioned link and in fact measures the importance of the link. In this paper, we compute the link centrality by closeness centrality which computes the sum of the distance of all possible paths contains

the mentioned link as shown in Eq. 6.

$$CEN = 1/\sum_{\forall u,\ u \neq n(e_v), n'(e_v)} d(permutation(\{n(e_v), n'(e_v)\}, u)) \quad (6)$$

In Eq. 6, $d(permutation(\{n(e_v), n'(e_v)\}, u))$is the distance between node u and two end nodes $n(e_v)$ and $n'(e_v)$of virtual link e_v. In fact, we want to compute the distance between other nodes from edge e_v. So permutation $\{n(e_v), n'(e_v)\}$, shows the mentioned link which the path should pass through that link.

After extracting these five features, the normalized values of features will be computed and a feature vector was made. The extracted feature vector is shown in Eq. 7.

$$FeatureVector = (P_v(e_v), AVG_{p_v}(e_v), DEG(e_v), CON(e_v), CEN(e_v)) \quad (7)$$

The feature vector is used in extracting features of the training and testing set and training the classifier and also in getting output from trained SOM-classifier.

II. Generating the training dataset

After designing a classifier, it should be trained and tested by trained and test dataset. We collect the training and testing dataset in two ways: (1) from generated uncertain virtual networks by GT-ITM tools and clustering them by pKwikCluster algorithm and extracting certain virtual networks and learning their features, and (2) from generated certain virtual networks by GT-ITM tools and adding noise to them and making uncertain virtual networks. We divide the extracted dataset to two sections; training and testing respectively with 80% and 20% data.

III. Training and testing

In the training process, the required parameters initialized and the SOM-classifier trained by training dataset in several epochs. The input of the training process is an uncertain virtual network and in each UVN, there are several uncertain links. For every uncertain link, the required features (*FeatureVector*) extracted. The output of this process is a trained SOM-classifier. SOM-classifier is a self-organizing map neural network which is unsupervised in real. But we apply the supervised method and train the classifier by labelled data. The supervised method, in each epoch, compare the output with the existed labels of data. The sum of squares error (SSE) is a metric to compare and the classifier is trained when its value is close to zero. SSE metric shown in Eq. 8.

$$SSE = \frac{1}{2}\sum_{i=1}^{n}(x_i - x')^2 \quad (8)$$

As shown in Eq. 8, suppose there are n output that should be compared with labeled data. Also, suppose x_i is an output of training SOM-classifier and x' is a label. We show the training process in Algorithm 5.

In Algorithm 5, the inputs are the SOM-classifier, a training set, number of epochs and learning rate. The output of this algorithm is trained SOM-classifier. At first, the algorithm extracts all features of edges in training set (rows 1-3). Then it repeats the training process (row 4) until the computed SSE reaches to zero number (row 13). In each iteration, all of the parameters are initialized (row 5) and then in some epoch (row 6), three phases of the SOM algorithm applied (rows 7-9) that will be explained. After all, epochs are run, algorithm get an output of trained SOM-classifier (row 11) and then compute SSE to consider the condition of the main loop (row 12).

These three phases are competition, cooperation and adaptation phase:

- **Competition phase:** in the competition phase, a winner neuron is selected (row 7) by assigning all inputs to SOM-classifier. In this paper, the extracted feature vector is one of the inputs. The other inputs are the number of epochs and learning rate. The winner neuron is selected based on the similarity between inputs and neurons. We used Euclidean distance as shown in the used equation in row 7. i is the Id number of winner neuron. $w_j(t)$ is the weight of neuron j at time t. The weight matrix is a matrix with cells same as SOM lattice size (We used grid lattice in this paper) and initializes with constant numbers and then changes to optimal values during the training process. Optimal values are values that train SOM-classifier with the training set. *FeatureVector* shows the feature vector of selected link edge (row 2).

Algorithm 5 Training Algorithm

Require: *input*: SOM-classifier, TrainingSet, numEpoch (Number of epochs), η (Learning rate);
output: Trained parameters in SOM-classifier;

1. **for** edge ∈ TraingSet **do**
2. FeatureVector = getFeatureVector(edge);
3. **end for**
4. **repeat**
5. Initialize all the parameters in SOM-classifier;
6. **while** iteration < numEpoch **do**
7. Competition phase: Compute winner neuron i using equation: $i = arg\,min_j \parallel FeatureVector - w_j(t) \parallel$;
8. Cooperation phase: Compute $h_{ij}(f_v, t)$ for all neurons (j) using: $h_{ij}(FeatureVector, t) = exp(-\frac{1}{2}\frac{d_{ij}^2}{\sigma(t)^2})$;
9. Adaptation phase: Compute WeightVectorj(t+1) using: $w_j(t+1) = w_j(t) + \sum_{f_v \epsilon FeatureVector} \Delta w_j(FeatureVector, t)$;
10. **end while**
11. $EdgeExistence = SOMclassifier.getOutput(FeatureVector)$;
12. compute SSE with equation: $\frac{1}{2}\sum_{i=1}^{|EdgeExistance|}(EdgeExistance_i - label(FeatureVector_i))$
13. **until** ($SSE \leq \epsilon$)
14. Return Trained parameters in SOM-classifier;

- **Cooperation phase:** in the cooperation phase, the amount of stimulation of winner neuron's neighbors should be computed (row 8). At this point, any neuron that has been won will also notify as many other neurons as possible. neighbors close to the winner neuron are more stimulated, and vice versa, other neurons are less stimulated. The number of stimulated neurons depends on the distance of the

winner neuron from other neurons and is calculated by an equation in row 8. $h_{ij}(FeatureV\ ector, t)$ is the amount of stimulation of neuron j against winner neuron *i*, when the input of SOM-classifier is vector $FeatureVector$ at time t. d_{ij} is Euclidean distance between neuron j and winner neuron *i*. In fact, the used function in the equation of row 8, is famous to Gaussian function.

- **Adaptation phase:** the algorithm initializes the weight matrix with the constant value before training process. In the adaptation phase, the weight matrix is updated. In fact, when a neuron won and the cooperation phase completed, in this phase, the weight of all lattice neurons was updated. Updating the weight matrix can be done sequentially (an updating phase runs with any input vector) or in batch processing manner (only one updating run with receiving all input vector). Used equation shown in Row 9. $w_j(t+1)$ is weight matrix neuron j at time t+1. $\Delta\ w_j(x,t)$ is weight changes for input x at time t that depends on learning rate η.

In the testing process, we apply the training set as input and compare the real output with SOM-classifier output. The testing algorithm is shown in Algorithm 6.

Algorithm 6 Testing Algorithm

Require: *input*: SOM-classifier, TestingSet;
output: validity indices;

1. Initialize all the parameters in SOM-classifier;
2. **for** edge ∈ TraingSet **do**
3. FeatureVector = getFeatureVector(edge);
4. **end for**
5. EdgeExistence=SOM-classifier.getOutput(FeatureVector);
6. **for** edge ∈ TestingSet **do**
7. **if** SOM-classifier answered correctly **then**
8. signal(SUCCESS);
9. **end if**
10. **end for**

As shown in Algorithm 6, the algorithm needs to another dataset as a testing set. In this regard, the collected dataset divided into training and testing dataset respectively 80% and 20%. The proposed algorithm initializes the parameters at first (row 1), for all requests in the dataset, extracts feature vector (rows 2-4) and finds the edge existence using SOM-classifier (row 5). Then, if the SOM-classifier is able to find edge existence successfully, the algorithm sends a success signal (rows 7-9).

B. Second step: clustering

After extracting a certain version of the uncertain virtual network, this step clusters the certain virtual network. As mentioned in this paper, we are considering uncertain virtual networks with probabilistic information on links. So, the requested topology is not important for users. In this regard, we can cluster the highly connected nodes to reduce the cost and increase the revenue. The highly connected nodes are a set of nodes with more than $|V\ |/2$ links as mentioned in section 3.5. In this step, the proposed algorithm uses HCS algorithm as a standard certain graph clustering algorithm.

C. Third step: virtual network embedding

After running two first steps, the requested uncertain virtual network converted to a compact certain virtual network and so, obviously, its mapping will require fewer resources. In this step, we used the standard virtual network embedding algorithm to embed the requested VN. We used the one-step embedding algorithm (EE-CTA [56]) and embed VNs in only one phase. In fact, one-step algorithm complete two phases of node mapping and link mapping in one phase concurrently and most of the evolutionary algorithms are in this category.

5- EXPERIMENTS and RESULTS

This section describes the experimental analysis of proposed UVNE and its three steps compared with other classifying, clustering and virtual network embedding algorithms. Ta 2 indices the compared indices in each step and all compared algorithms.

TABLE 2
Used indices and compared algorithms in each step of UVNE

Steps	Indices	Compared algorithms
(1) Edge Prediction	Error percentage Validation accuracy Testing accuracy SOM-classifier accuracy	SVM [58] ANN-1 ANN-2 Threshold-based
(2) Clustering	DBI DI SC	SVM [58] pKwikCluster [54] ANN-2
(3) VNE	Cost Revenue Acceptance ratio Node and link utilization	Robust optimization [59] VCDN [60] Threshold-based GAT-DRL [16] Ce-VNE (GCN-based) [25]

As shown in Table 2, we compare proposed SOM-classifier in edge prediction step with other four classifier algorithm based on error percentage and accuracy. The second step compared with traditional pKwikCluster, SVM and ANN-2 algorithm based on DBI, DI and SC indices. At last, the third step compared with three related embedding algorithms and recent advanced approaches based on cost, revenue and acceptance ratio. ANN-2 is a traditional artificial neural network with three neurons in the input layer, 10 neurons in the hidden layer neurons and two output layer neurons (binary classifier). ANN-1 has three, three, and two neurons in the input layer, hidden layer and output layer, respectively.

TABLE 3
Used parameter in experiment of UVNE

Parameter	Value
Classifier	SOM
SOM-lattice	Grid
SOM-similarity	Gaussian
SOM-learning method	Kohonen
Learning rate (η)	0.8
Number of Epochs	50
Clustering threshold	0.5
Substrate network topology	BCube [57]
Nodes in substrate network	500
Links in substrate network	1128
Virtual network topology	[Ring, Star, Random]
Nodes in virtual networks	3-10
Links in virtual networks	5-30

All used parameters in experimental evaluation listed in Table 3. All the experiments are compared in an operating system with an Intel Corei7- 8550 U- 1.89 GHz processor with a GPU 4 GB and 12 GB Ram and Windows 10 Enterprise. In generating training and testing dataset, we use GT-ITM [55] tools and generate 100 certain virtual networks and 100 uncertain virtual networks with three topologies of ring, star, and random. As mentioned, the training and testing set generated in two ways:

1. Adding noise to certain virtual networks which were generated with GT-ITM tools and making uncertain virtual networks (NoiseDataSet).
2. Using pKwikCluster algorithm to generating certain virtual networks of uncertain virtual networks which were generated with GT-ITM tools (pKwikDataSet).

To evaluate the proposed UVNE algorithm in a context resembling Software-Defined Networking, our simulation environment was designed to incorporate key characteristics of an SDN architecture. Most importantly, we assume the presence of a centralized controller that maintains a global and perfect view of the substrate network state. This means that our embedding algorithm has instantaneous access to the exact available CPU capacity of all substrate nodes and the available bandwidth of all substrate links at the time of each embedding decision. This ideal global knowledge eliminates the inaccuracies that might arise from distributed or outdated network state information, allowing us to isolate and measure the pure performance benefits of the UVNE algorithm itself. The ability to leverage this global view is a fundamental advantage offered by the SDN paradigm, and our simulation setup accurately reflects this capability. After a successful embedding, the cost and resource utilization are calculated based on this global information.

A. Edge prediction with SOM-classifier

This section describes all experiments to compare the proposed SOM-classifier based on error percentage and classifier accuracy. In fact, each classifier should be able to predicate the right label for inputs. In this paper, the classifier is a binary classifier and include only two output/label for each input. So, the proposed algorithm should be able to predicate the existence of edge in an uncertain virtual network and labels an uncertain link with one or zero (the value of one, shows that the mentioned link should be as a certain link in UVN and the value of zero, shows the uncertain link is not required in the UVN lifetime.). In the first test, we compare error percentage on two generated datasets NoiseDataSet and pKwikDataSet and the results are shown in Figure 2 and 3 respectively.

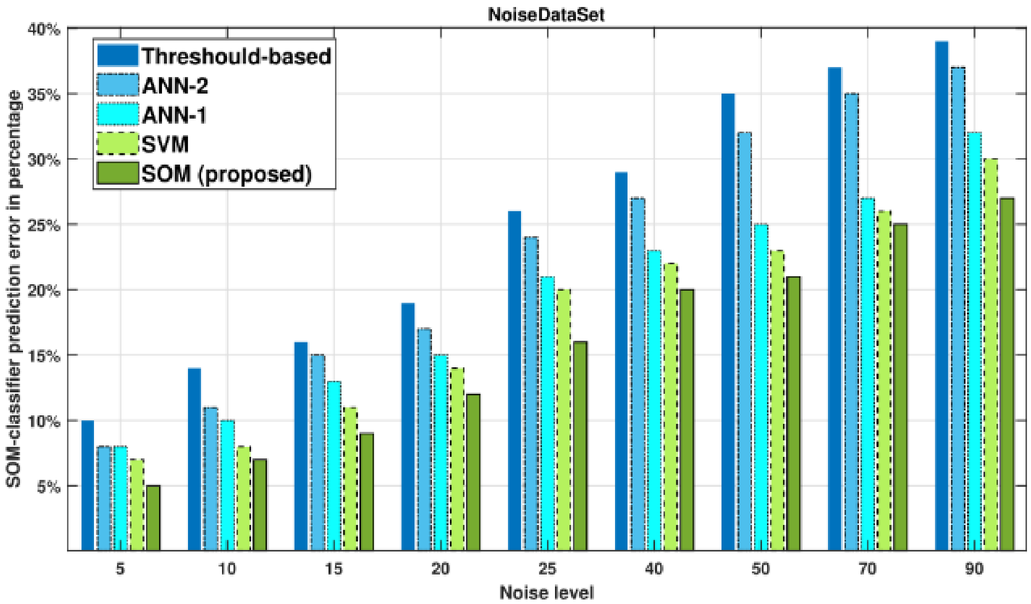


Figure 2. Error percentage of proposed SOM-classifier based on noise level in dataset NoiseDataSet

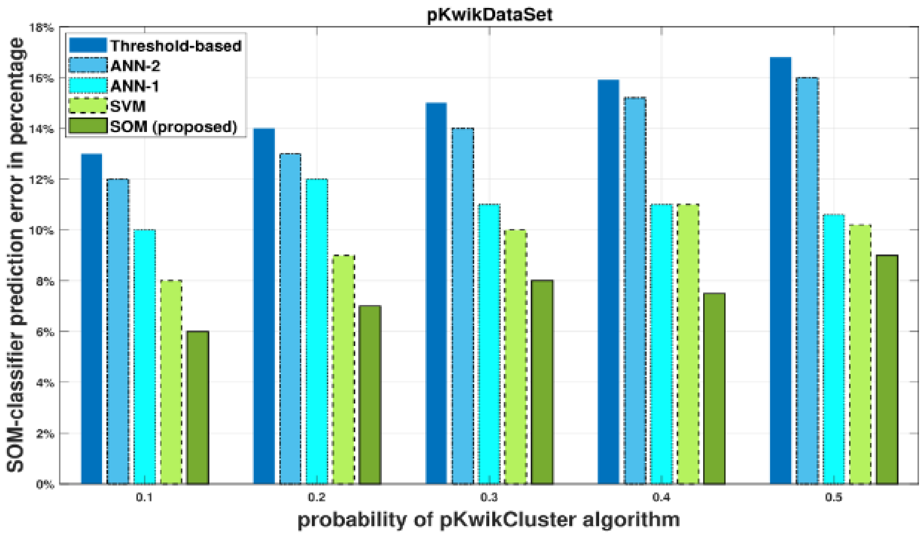


Figure 3. Error percentage of proposed SOM-classifier based on noise level in dataset pKwikDataSet

As shown in Figure 2, we generate a datasets with various noise level and use the generated datasets to compute error percentage in proposed SOM-classifier and three state of the artworks. SVM is one of the related works which is proposed for edge prediction in large uncertain graphs. ANN-1 and ANN-2 are two related works based on standard neural network and Threshold-based is a traditional algorithm which predicates the edges based on their probability. This test indicates, the proposed SOM- classifier has a low error percentage in comparison with others. The error percentage starts from about 5% when the noise level is 5% and increases up to 25% with 90% noise. Adding 90% noise to dataset makes all algorithms unsuitable, because this level of noise, eliminates all features of network and algorithms cannot predicate the edges existence.

As shown in Figure 3, the same test has been run on generated pKwikDataSet and the results show the proposed SOM-classifier gained less error percentage compared with the other three works. In the second test, we compare validation and testing accuracy in proposed and related works. The results are shown in Table 4. In this test, the noise level of NoiseDataSet was 0.5 and our dataset had 20% of whole generated dataset.

TABLE 4
Validation and testing accuracy of proposed SOM-classifier for edge prediction

Algorithms	**NoiseDataSet**		**pKwikDataSet**	
	Validation accuracy	Testing accuracy	Validation accuracy	Testing accuracy
SOM	97%	93%	96%	91%
SVM	95%	84%	94%	82%
ANN-1	75%	663%	73%	66%
ANN-2	70%	8%	69%	62%
Threshold-based	65%	48%	63%	47%

As shown in Table 4, the validation and testing accuracy in proposed SOM- classifier is more than other works. This is due to the use of neighborhood information in the proposed method for prediction of the edges. As shown the validation accuracy is more than testing accuracy. After proposed SOM-classifier, SVM gained high accuracy due to the use of only one neighborhood information (neighbor links probability). ANN-1, ANN-2 and Threshold-based approaches have respectively low accuracy. The traditional neural networks do not use any neighborhood information and were trained unsupervised. So, their accuracy is less than others. At last, the Threshold-based approach has less accuracy because it predicates the edges existence only based on their probability.

TABLE 5
SOM-classifier accuracy for edge prediction

Dataset	**SOM**	**SVM**	**ANN-1**	**ANN-2**	**Threshold-based**
NoiseDataSet	92%	90%	79%	71%	66%
pKwikDataSet	91%	88%	75%	69%	64%

B. Clustering

This section compares the clustering algorithm which apply on extracted CVNs with related works based on three indices DBI, DI and SC. Davies-Bouldin index (DBI) is a clustering index and computes how well the clustering has been done. This index is shown in Eq. 9. Where k is the number of clusters, σ_x is the average distance of all nodes in a cluster from the central node and $d(c_i, c_j)$ is the distance between two centers of cluster c_i and c_j.

$$DBI = \frac{1}{k}\sum_{i=1}^{k} max_{i\neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \qquad (9)$$

Dunn index (DI) index is an internal cluster evaluating metric which is shown in Eq. 10. If there are k clusters, then the Dunn Index for the set is defined as Eq. 10. Where Δx is maximum d(x, y) with x and y from mentioned cluster.

$$DI = \frac{min_{1\leq i\leq j\leq k} d(c_i, c_j)}{max_{1\leq x\leq k}\Delta x} \tag{10}$$

Silhouette coefficient (SC) computes how well each object lies within its cluster which is shown in Eq. 11 and 12. Where a(i) is the average dissimilarity of node i from the other nodes within the cluster and b(i) is the minimum average dissimilarity of node i to the nodes of all other clusters.

$$SC = \frac{1}{k}\sum_{i=1}^{k} s(i) \tag{11}$$

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(j)\}} \tag{12}$$

The results of comparing clustering algorithm and three related works are based on two generated datasets are shown in Figure 4. As demonstrated, the proposed clustering algorithm in the second step of the proposed method (SOM+HCS) reaches low DBI compared with ANN-2, pKwikCluster and SVM based algorithm. DBI shows the clustering power to the number of clusters and a low number of this index makes a better clustering algorithm. The SVM based algorithm has low DBI compared with pKwikCluster and ANN-2, because uses neighborhood information to train the SVM machine. pKwikCluster algorithm has the third rank in clustering based on DBI index and the last one is ANN-2. In DI and SC indices, a high number of the index has shown the better clustering algorithm and the proposed SOM+HCS has high value compared with other algorithms, because of using of five neighborhood information in training process and using HCS algorithm in clustering step.

C. Virtual network embedding

This section compares the proposed UVNE algorithm with Robust, VCDN and threshold-based algorithms based on cost, revenue, resource utilization and execution time. Cost and revenue are the main indices in the evaluation of embedding algorithms. Cost is an index which shows the sum of assigned nodes' capacity and links' bandwidth for a requested virtual network and computes by Eq. 13.

$$\begin{aligned} Cost(G_v) &= Cost(N_v) + Cost(E_v) \\ &= \sum_{n_v\in N_v} C_v(n_v) \times Y_{n_s}^{n_v} \\ &+ \sum_{e_v\in E_v} B_v(e_v) \times X_{e_s}^{e_v} \\ &\times |P(e_v)| \end{aligned} \tag{13}$$

In Eq. 13, for a requested UVN G_v, embedding cost is equal with sum of needed node capacity and link bandwidth which is respectively $Cost(N_v)$ and $Cost(E_v)$. Revenue is another index in virtual network embedding which is computed by Eq. 14 and shows the sum of requested resources for each uncertain virtual network.

$$Rev(G_v) = \sum_{n_v\in N_v} C_v(n_v) + \sum_{e_v\in E_v} B_v(e_v) \tag{14}$$

The acceptance ratio is another index which shows the number of accepted virtual networks in each time interval. Also, node and link utilization depend on the percentage of used resources to available resources in each time interval and for each server in data center. In this paper, the first test computes the cost of embedding and the results are shown in Figure 5.

As shown in Figure 5, the proposed UVNE achieves a lower average cost compared to other algorithms. This efficiency is partly attributed to the clustering step, which is empowered by the global network view available in an SDN-like setting. The algorithm can make informed decisions about consolidating virtual nodes onto physically proximate substrate nodes, thereby reducing costly long-distance link mappings.

\

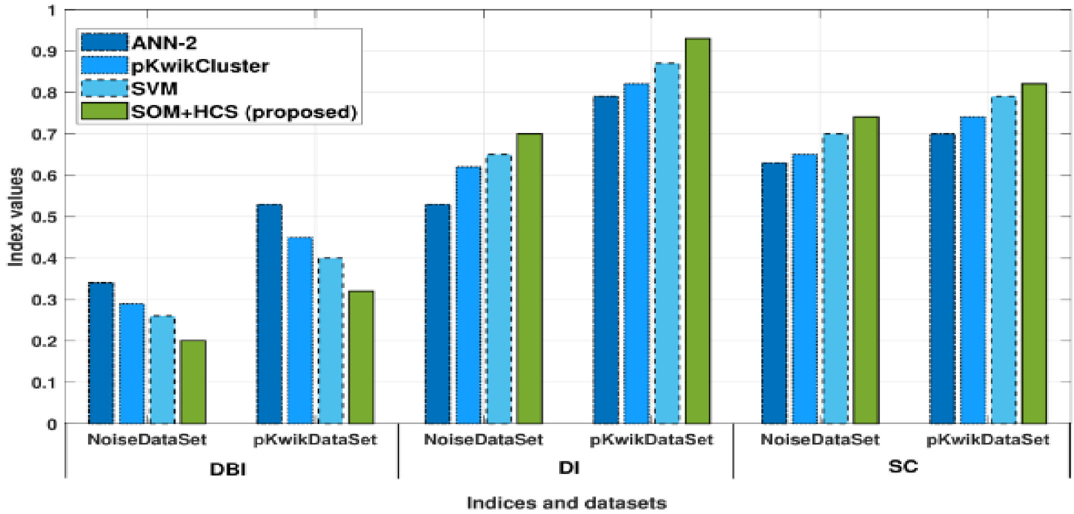


Figure 4. DBI, DI and SC comparison based on two datasets

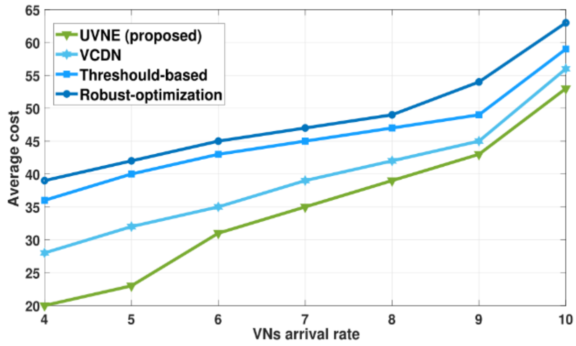


Figure 5. Average cost based on virtual networks arrival rate

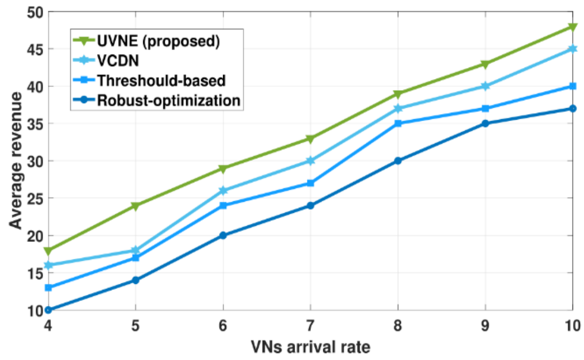


Figure 6. Average revenue based on virtual networks arrival rate

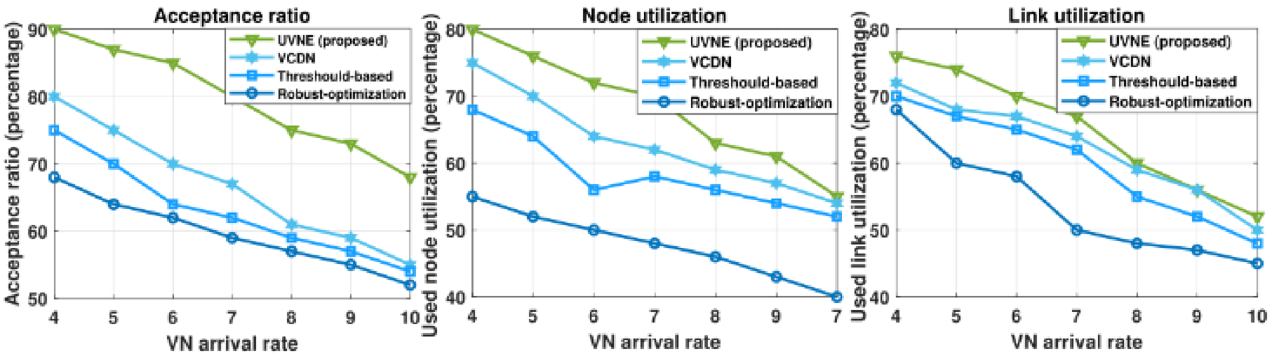


Figure 7. Acceptance ratio, node and link utilization of proposed UVNE based on virtual networks arrival rate

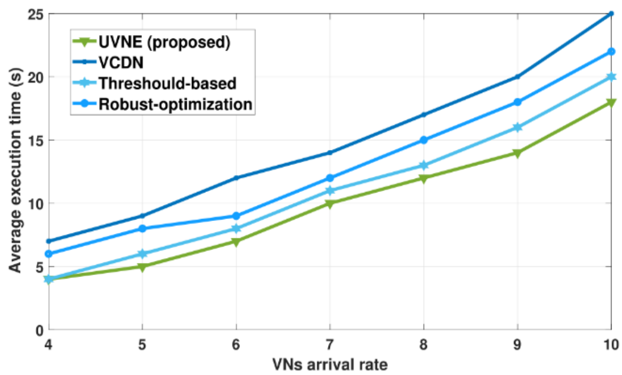


Figure 8. Average execution time of proposed UVNE based on virtual networks arrival rate

The second test compares reached revenue for InP which is shown in Figure 6. As demonstrated the proposed UVNE algorithm has high average revenue compared with others. The Robust-optimization algorithm has low revenue, because of solving the worst case of embedding and has low acceptance ratio.

The third and forth tests compute acceptance ratio and node and link utilization respectively and the results are shown in Figure 7. As shown the proposed UVNE has high acceptance ratio, high node utilization and also high link utilization compared with the other three algorithms.

The last test computes the execution time of the proposed UVNE algorithm. The execution time of VCDN algorithm is higher than others because it considers the sudden changes of the requested virtual network and tries to answer to changed requests. The proposed UVNE has low execution time because of using a SOM neural network and train it based on neighborhood information.

D. Impact of Substrate Network Knowledge Accuracy

To underscore the importance of the global view provided by an SDN controller, we conducted an additional experiment comparing UVNE's performance under two scenarios: (1) Perfect Global Knowledge (simulating an ideal SDN controller) and (2) Partial/Delayed Knowledge (simulating a non-SDN environment with outdated information). The results, summarized in Table 6, clearly show that when the algorithm operates with imperfect information, the acceptance ratio decreases by approximately 15% and the cost increases by 10% due to embedding failures and suboptimal mappings. This experiment validates that

the performance advantages of UVNE are fully realized when deployed within an SDN framework that provides accurate and timely network state information.

TABLE 6
Performance comparison based on substrate network knowledge accuracy

Scenario	Average Cost	Acceptance Ratio
Perfect Global Knowledge (SDN)	145	89%
Partial/Delayed Knowledge	160	76%

6- CONCLUSION and FUTURE WORK

In this paper, we proposed an uncertain virtual network embedding algorithm which requested virtual networks had non-deterministic information on links based on edge prediction. Uncertain virtual networks introduced in this paper and formulated by an uncertain graph. The proposed UVNE algorithm has three steps; the first step selects the suitable certain virtual network by prediction edge existence based on neighbor information. the second step clusters the selected CVN and computes a compact one. At last, in the third step, the CVN is embedded on a substrate network with one-step embedding algorithms.

We tested our proposed UVNE on generated datasets and proposed SOM-classifier algorithm on real networks. Our experimental evaluation demonstrates proposed SOM- classifier reach high DBI, DI and SC metrics. Also, proposed UVNE increases InPs revenue and acceptance ratio and decreases user's cost.

Further research is needed to consider the design of clustering large uncertain virtual networks. This work can also be extended by parallel implementation of the proposed algorithm on a high-performance computing cluster. Also, uncertain virtual network embedding problem can be solved by extracting association rules and association rules mining algorithms. We consider uncertain information on links in VNs. In future work, the uncertainty can be considered on both nodes and links.

Conflict of Interest

The author declares that she has no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] G. Sun, D. Liao, D. Zhao, Z. Sun, & V. Chang. (2018). Towards provisioning hybrid virtual networks in federated cloud data centers. *Future Generation Computer Systems*. [Online]. *87*, pp. 457–469. Available: https://doi.org/10.1016/j.future.2017.09.065

[2] A. Jahani, "Virtual network embedding based on univariate distribution estimation," In Proceeding 11th Internatinal Conference Computer Engineering and Knowledge (ICCKE), 2021, pp. 284–289. [Online]. Available: https://doi.org/10.1109/ICCKE54056.2021.9721444

[3] A. Jahani, L. M. Khanli, M. T. Hagh, & M. A. Badamchizadeh. (2019). Green virtual network embedding with supervised self-organizing map. *Neurocomputing*. [Online]. *351*, pp. 60–76. Available: https://doi.org/10.1016/j.neucom.2019.04.090

[4] C. Aguilar-Fuster & J. Rubio-Loyola. (2021). A novel evaluation function for higher acceptance rates and more profitable metaheuristic-based online virtual network embedding. *Computer Networks*. [Online]. *195*, p. 108191. Available: https://doi.org/10.1016/j.comnet.2021.108191

[5] C. Jiang & P. Zhang. (2021). Incorporating energy and load balance into virtual network embedding process. *Springer*. [Online]. *129*, pp. 245–268. Available: https://doi.org/10.1016/j.comcom.2018.07.027

[6] A. Jahani and L. M. Khanli, "CATA-VN: Coordinated and topology-aware virtual network service provisioning in data centers network," in *2017 7th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, Iran, 2017, pp. 353–358. https://doi.org/10.1109/ICCKE.2017.8167904

[7] B. A. Souza, G. R. Mateus, & F. S. Souza. (2018). Compact and extended formulations for the virtual network embedding problem. *Electronic Notes in Discrete Mathematics*. [Online]. *64*, pp. 205–214. Available: https://doi.org/10.1016/j.endm.2018.01.022

[8] S. R. Chowdhury, R. Ahmed, M. M. A. Khan, N. Shahriar, R. Boutaba, J. Mitra, & F. Zeng. (2016). Dedicated protection for survivable virtual network embedding. *IEEE Transactions on Network and Service Management*. [Online]. *13(4)*, pp. 913–926. Available: https://doi.org/10.1109/TNSM.2016.2574239

[9] H. Lei, T. Zhang, Y. Liu, Y. Zha, & X. Zhu. (2015). SGEESS: Smart green energy-efficient scheduling strategy with dynamic electricity price for data center. *Journal of Systems and Software*. [Online]. *108*, pp. 23–38. Available: https://doi.org/10.1016/j.jss.2015.06.026

[10] N. Triki, N. Kara, M. El Barachi, & S. Hadjres. (2015). A green energy-aware hybrid virtual network embedding approach. *Computer Networks*. [Online]. *91*, pp. 712–737. Available: https://doi.org/10.1016/j.comnet.2015.08.016

[11] D. Arnone, A. Barberi, D. La Cascia, E. R. Sanseverino, & G. Zizzo. (2017). Green data centres integration in smart grids: New frontiers for ancillary service provision. *Electric Power Systems Research*. [Online]. *148*, pp. 59–73. Available: https://doi.org/10.1016/j.epsr.2017.03.017

[12] M. A. Z. Soltani, S. A. H. Seno, & A. Mohajerzadeh. (2025). Optimizing SDN resource allocation using fuzzy logic and VM mapping technique. *Computing*. [Online]. *107(1)*. Available:

https://doi.org/10.1007/s00607-024-01360-4

[13] M. M. A. Khan, N. Shahriar, R. Ahmed, & R. Boutaba. (2016). Multi-path link embedding for survivability in virtual networks. *IEEE Transactions on Network and Service Management*. [Online]. *13(2)*, pp. 253–266. Available: https://doi.org/10.1109/TNSM.2016.2558598

[14] W. Wang, Y. Zhao, R. He, X. Yu, J. Zhang, H. Zheng, Y. Lin, & J. Han. (2016). Continuity-aware spectrum allocation schemes for virtual optical network embedding in elastic optical networks. *Optical Fiber Technology*. [Online]. *29*, pp. 28–33. Available: https://doi.org/10.1016/j.yofte.2016.01.008

[15] M. He, L. Zhuang, S. Tian, G. Wang, & K. Zhang. (2020). DROi: Energy-efficient virtual network embedding algorithm based on dynamic regions of interest. *Computer Networks*. [Online]. *166*, p. 106952. Available: https://doi.org/10.1016/j.comnet.2019.106952

[16] I. Ullah, Q. Ali, M. Ashraf, and Y.-H. Han, "Advanced virtual network embedding: Combining graph attention network and DRL for optimal resource utilization," in *2025 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, South Korea, 2025, pp. 550–555. Available: https://doi.org/10.1109/ICAIIC64266.2025.10920877

[17] M. Mangili, F. Martignon, & A. Capone. (2016). Performance analysis of content-centric and content-delivery networks with evolving object popularity. *Computer Networks*. [Online]. *94*, pp. 80–98. Available: https://doi.org/10.1016/j.comnet.2015.11.019

[18] K. Zhan, N. Chen, S. V. N. S. Kumar, G. Kibalya, P. Zhang, & H. Zhang. (2025). Edge computing network resource allocation based on virtual network embedding. *International Journal of Communication Systems*. [Online]. *38(1)*. Available: https://doi.org/10.1002/dac.5344

[19] F. Esposito, D. D. Paola, & I. Matta. (2016). On distributed virtual network embedding with guarantees. *IEEE/ACM Transactions on Networking*. [Online]. *24(1)*, pp. 569–582. Available: https://doi.org/10.1109/TNET.2014.2375826

[20] Y. Wang, Q. Hu, L. Nguyen, & M. Jalalitabar. (2023). Minimum-cost embedding of virtual networks: An iterative decomposition approach. *Computer Networks*. [Online]. *234*, p. 109907. Available: https://doi.org/10.1016/j.comnet.2023.109907

[21] G. Sun, H. Yu, L. Li, V. Anand, Y. Cai, & H. Di. (2012). Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter. *Photonic Network Communications*. [Online]. *23(2)*, pp. 109–122. Available: https://doi.org/10.1007/s11107-011-0341-z

[22] D. Jian, H. Tao, W. Jian, W. Hu, L. Jiang, & Y. Li. (2015). Virtual network embedding through node connectivity. *The Journal of China Universities of Posts and Telecommunications*. [Online]. *22(1)*, pp. 17–56. Available: https://doi.org/10.1016/S1005-8885(15)60620-3

[23] C. Wang, G. Liu, and Y. Yuan, "A novel method for virtual network embedding with incentive convergence mechanism," in *Third International Conference on Advanced Cloud and Big Data (CBD)*, Yangzhou, China, 2015, pp. 275–281. Available: https://doi.org/10.1109/CBD.2015.51

[24] S. Zhang, J. Wu, and S. Lu, "Virtual network embedding with substrate support for parallelization," in IEEE Global Communications Conference (GLOBECOM), 2013, pp. 2615–2620. [Online]. Available: https://doi.org/10.1109/GLOCOM.2012.6503544

[25] P. Zhang, Z. Luo, N. Kumar, M. Guizani, H. Zhang, & J. Wang. (2024). CE-VNE: Constraint escalation virtual network embedding algorithm assisted by graph convolutional networks. *Journal of Network and Computer Applications*. [Online]. *221*, p. 103736. Available: https://doi.org/10.1016/j.jnca.2023.103736

[26] L. Gong, Y. Wen, Z. Zhu, and T. Lee, "Toward profit-seeking virtual network embedding algorithm via global resource capacity," in *Proceeding IEEE INFOCOM*, 2014, pp. 1–9. Available: https://doi.org/10.1109/INFOCOM.2014.6847918

[27] T. Wang & M. Hamdi. (2016). Presto: Towards efficient online virtual network embedding in virtualized cloud data centers. *Computer Networks*. [Online]. *106*, pp. 196–208. Available: https://doi.org/10.1016/j.comnet.2016.06.036

[28] S. Haeri & L. Trajković. (2017). Virtual network embedding via Monte Carlo tree search. *IEEE Transactions on Cybernetics*. [Online]. *48*, pp. 510–521. Available: https://doi.org/10.1109/TCYB.2016.2645123

[29] M. T. Beck, A. Fischer, J. F. Botero, C. Linnhoff-Popien, & H. de Meer. (2015). Distributed and scalable embedding of virtual networks. *Journal of Network and Computer Applications*. [Online]. *56*, pp. 124–136. Available: https://doi.org/10.1016/j.jnca.2015.06.012

[30] J. F. Botero & X. Hesselbach. (2013). Greener networking in a network virtualization environment. *Computer Networks*. [Online]. *57(9)*, pp. 2021–2039. Available: https://doi.org/10.1016/j.comnet.2013.04.004

[31] X. Guan, B. Y. Choi, & S. Song. (2015). Energy-efficient virtual network embedding for green data centers using data center topology and future migration. *Computer Communications*. [Online]. *69*, pp. 50–59. Available: https://doi.org/10.1016/j.comcom.2015.05.003

[32] R. R. Oliveira, D. S. Marcon, L. R. Bays, M. C. Neves, L. P. Gaspary, D. Medhi, & M. P. Barcellos. (2015). Opportunistic resilience embedding (ORE):

Toward cost-efficient resilient virtual networks. *Computer Networks*. [Online]. *89*, pp. 59–77. Available: https://doi.org/10.1016/j.comnet.2015.07.010

[33] M. Chowdhury, M. R. Rahman, & R. Boutaba. (2012). Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking*. [Online]. *20(1)*, pp. 206–219. Available: https://doi.org/10.1109/TNET.2011.2159308

[34] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, & J. Wang. (2011). Virtual network embedding through topology-aware node ranking. *ACM SIGCOMM Computer Communication Review*. [Online]. *41(2)*, pp. 38–47. Available: https://doi.org/10.1145/1971162.1971168

[35] Z. Zhang, X. Cheng, S. Su, Y. Wang, K. Shuang, & Y. Luo. (2013). A unified enhanced particle swarm optimization-based virtual network embedding algorithm. *International Journal of Communication Systems*. [Online]. *26(8)*, pp. 1054–1073. Available: https://doi.org/10.1002/dac.1399

[36] X. Hesselbach, J. R. Amazonas, S. Villanueva, & J. F. Botero. (2016). Coordinated node and link mapping VNE using a new paths algebra strategy. *Journal of Network and Computer Applications*. [Online]. *69*, pp. 14–26. Available: https://doi.org/10.1016/j.jnca.2016.02.025

[37] N. Ogino, T. Kitahara, S. Arakawa, & M. Murata. (2017). Virtual network embedding with multiple priority classes sharing substrate resources. *Computer Networks*. [Online]. *112*, pp. 52–66. Available: https://doi.org/10.1016/j.comnet.2016.10.007

[38] M. Bienkowski, A. Feldmann, J. Graßler, G. Schaffrath, & S. Schmid. (2014). The wide-area virtual service migration problem: A competitive analysis approach. *IEEE/ACM Transactions on Networking*. [Online]. *22(1)*, pp. 165–178. Available: https://doi.org/10.1109/TNET.2013.2245676

[39] N. F. Butt, M. Chowdhury, and R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," in *International Conference on Research in Networking (NETWORKING)*, Chennai, India, 2010, pp. 27–39. Available: https://doi.org/10.1007/978-3-642-12963-6_3

[40] O. Soualah, I. Fajjari, M. Hadji, N. Aitsaadi, and D. Zeghlache, "A novel virtual network embedding scheme based on Gomory-Hu tree within cloud's backbone," in *IEEE/IFIP Network Operations and Management Symposium (NOMS)*, Istanbul, Turkey, 2016, pp. 536–542. https://doi.org/10.1109/NOMS.2016.7502855

[41] S. R. Zahedi, S. Jamali, & P. Bayat. (2022). EmcFIS: Evolutionary multi-criteria fuzzy inference system for virtual network function placement and routing. *Applied Soft Computing*. [Online]. *117*, p. 108427. Available: https://doi.org/10.1016/j.asoc.2021.108427

[42] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in Proceedings of the *1st ACM Workshop on Virtualized Infrastructure Systems and Architectures (VISA)*, Barcelona, Spain, 2009, pp. 81–88. https://doi.org/10.1145/1592648.1592662

[43] I. Houidi, W. Louati, W. B. Ameur, & D. Zeghlache. (2011). Virtual network provisioning across multiple substrate networks. *Computer Networks*. [Online]. *55(4)*, pp. 1011–1023. Available: https://doi.org/10.1016/j.comnet.2010.12.011

[44] I. Fajjari, N. A. Saadi, G. Pujolle, and H. Zimmermann, "VNE-AC: Virtual network embedding algorithm based on ant colony metaheuristic," in *IEEE International Conference on Communications (ICC)*, Kyoto, Japan, 2011, pp. 1–6. Available: https://doi.org/10.1109/icc.2011.5963442

[45] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, & H. de Meer. (2012). Energy efficient virtual network embedding. *IEEE Communications Letters*. [Online]. *16(5)*, pp. 756–759. Available: https://doi.org/10.1109/LCOMM.2012.030912.120082

[46] A. Pages, J. Perelló, S. Spadaro, & G. Junyent. (2012). Strategies for virtual optical network allocation. *IEEE Communications Letters*. [Online]. *16(2)*, pp. 268–271. Available: https://doi.org/10.1109/LCOMM.2011.120211.111900

[47] H. Yu, V. Anand, C. Qiao, H. Di, & X. Wei. (2012). A cost-efficient design of virtual infrastructures with joint node and link mapping. *Journal of Network and Systems Management*. [Online]. *20(1)*, pp. 97–115. Available: https://doi.org/10.1007/s10922-011-9209-x

[48] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, & J. Wang. (2012). Virtual network embedding through topology awareness and optimization. *Computer Networks*. [Online]. *56(6)*, pp. 1797–1813. Available: https://doi.org/10.1016/j.comnet.2012.01.022

[49] H. Di, H. Yu, V. Anand, L. Li, G. Sun, & B. Dong. (2012). Efficient online virtual network mapping using resource evaluation. *Journal of Network and Systems Management*. [Online]. *20(4)*, pp. 468–488. Available: https://doi.org/10.1007/s10922-012-9249-x

[50] C. Papagianni, A. Leivadeas, S. Papavassiliou, V. Maglaris, C. Cervelló-Pastor, & A. Monje. (2013). On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers*. [Online]. *62(6)*, pp. 1060–1071. Available: https://doi.org/10.1109/TC.2013.31

[51] F. Zhu & H. Wang. (2016). A modified ACO algorithm for virtual network embedding based on graph decomposition. *Computer Communications*. [Online]. *80*, pp. 1–15. Available: https://doi.org/10.1016/j.comcom.2015.07.014

[52] G. Kollios, M. Potamias, & E. Terzi. (2013). Clustering large probabilistic graphs. *IEEE*

Transactions on Knowledge and Data Engineering. [Online]. *25(2)*, pp. 325–336. Available: https://doi.org/10.1109/TKDE.2011.243

[53] N. Ailon, M. Charikar, & A. Newman. (2008). Aggregating inconsistent information: Ranking and clustering. *Journal of the ACM (JACM)*. [Online]. *55(5)*, Article 23. Available: https://doi.org/10.1145/1411509.1411513

[54] K. K. TG, S. Tomar, S. K. Addya, A. Satpathy, & S. G. Koolagudi. (2024). EFRAS: Emulated framework to develop and analyze dynamic virtual network embedding strategies over SDN infrastructure. *Simulation Modelling Practice and Theory*. [Online]. *134*, p. 102952. Available: https://doi.org/10.1016/j.simpat.2024.102952

[55] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *INFOCOM'96: Fifteenth Annual Joint Conference of the IEEE Computer Societies – Networking the Next Generation*, San Francisco, CA, USA, 1996, vol. 2, pp. 594–602. https://doi.org/10.1109/INFCOM.1996.493353

[56] A. Jahani, L. M. Khanli, M. T. Hagh, & M. A. Badamchizadeh. (2019). EECTA: Energy efficient, concurrent and topology-aware virtual network embedding as a multi-objective optimization problem. *Computer Standards & Interfaces*. [Online]. *66*, p. 103351. Available: https://doi.org/10.1016/j.csi.2019.04.010

[57] C. Guo, G. Lu, D. Li, H. Wu, X. Zhang, Y. Shi, C. Tian, Y. Zhang, & S. Lu. (2009). BCube: A high performance, server-centric network architecture for modular data centers. *ACM SIGCOMM Computer Communication Review*. [Online]. *39(4)*, pp. 63–74. Available:https://doi.org/10.1145/1592568.1592577